CLIC: an agent-based interactive and autonomous piece of art

Laurent Lacomme, Yves Demazeau and Julie Dugdale

Laboratoire d'Informatique de Grenoble – Grenoble INP, CNRS, Université Pierre Mendès France – {Laurent.Lacomme,Yves.Demazeau,Julie.Dugdale}@imag.fr

Abstract This work consists of integrating programming paradigms such as multi-agent systems and rule-based reasoning into a multimedia creation and display platform for interactive artistic creation. It has been developed in order to allow artists to build dynamic and interactive exhibitions based on pictures and sounds and featuring self-evolving and autonomous configurations.

1 Introduction

The project we present here is part of the biennial event called "Rencontres-i", organized by the Grenoble theatre Hexagone about art and science collaboration. Inspired by the behavior of social insects, the theme of this year's event is *gathering and swarming*. The event involves the participation of the MAGMA multi-agent systems team and a group of artists named Coincoin Production (www.collectif-coin.com) that has jointly coproduced the work. This collaborative project aims at using scientific work to produce a piece of art, matching the theme of the event, which represents the artists' views. This paper describes the scientific aspect of it.

The artistic idea for this work consists of presenting, reorganizing and developing the images and sounds that can be found on a place that is well-known to the visitors: the university campus. The result was a dynamic and interactive exhibition named CLIC – Conception d'un Logiciel Interactif Collaboratif (Conception of a collaborative and interactive software), which was shown to the public for three weeks. Thus, the computer science part of the work was to design and develop a software system, on top of the *Max/MSP/Jitter* media platform that is used by the artists, in order to design and build a multimedia exhibition. The goal was not only to build a system that can be used for this specific project, but to design a flexible and easy-to-use software that can be used by different artists to construct other multimedia exhibitions. This software has taken the form of a multi-agent system, where agents linked to media interact in several environments to produce commands for the multimedia platform.

In the following section we firstly discuss the related works and the specificity of the project. The theoretical analysis of the artistic challenge and the design of a solution are presented in section three. Section four covers the practical problems that have appeared during the development as well as the solutions we have found. Section five describes the final software and the exhibition. Finally, we conclude with an evaluation of the work and draw some perspectives.

2 Related works

Several works have already been achieved concerning the collaboration between art and artificial intelligence. From the beginning of AI, and the first expert systems, projects have been exploring the relation between artistic creation and the possibilities given by AI systems. One of the first projects was AARON [1], an expert system painter, which was able to create original drawings that represented scenes that were understandable and appreciated by humans. Although this project involved a creative artificial intelligence, the goal was to make it autonomously create finalized pieces of art. The project we are concerned with consists, on the contrary, in creating a dynamic – thus never completed – exhibition.

Projects that use AI paradigms such as cognitive reasoning and multi-agent systems for developing dynamic and interactive media production already exist. One of the most significant works is a series of dynamic self-evolving paintings named "Le Jardin des Hasards" [5], whose paintings are composed of several agent elements, which evolve with regard to the public activity and meteorological data, in a biological-inspired manner. Another significant work is a project named "M@trice @ctive" [4], which consisted of representing a painting from Kandinsky as a 3D environment in which shape-elements evolve and move according to initial conditions and interaction laws inspired from the painter's description of his own work; the visitor can navigate in this environment to discover new points of view on the painting.

Our project is different from these ones in the sense that it does not consist of modeling specific relationships between elements. Instead, it aims at providing a flexible, reusable platform for artists to create several dynamic and interactive exhibitions. It has also the goal of being easy to use by artists, without any help from computer scientists, whereas the above projects needed to be directly configured by computer scientists.

3 Theoretical analysis

The artists wanted to build an exhibition that would be autonomous and dynamic. It means that the exhibition, controlled by a computer, was going to evolve by itself through time, without any direct action from the artists. Hence the main goal

was to provide a *self-evolving*, *autonomous* and *dynamic* software component, whose behavior and evolution should be both scripted and unpredictable. Thus, the artists had to be able to set a global behavior for their exhibition, in order to express their creative views, but the precise evolution of the exhibition, such as the choice of one picture or another, or the display for a particular effect, should be determined by the software over time.

A good way to achieve this was to incorporate some *artificial intelligence* capabilities, so that the system can handle different situations and adapt its behavior, following goals or rules that reflect the artists' expectations. That would guarantee both variability – because of the non-algorithmic aspect in the software design – and stability – because of the ability of the software behavior to be matched to artists' wishes. That solution would also ensure another important property of the software: interactivity. Indeed, as the behavior of the exhibition is computed in real-time by the program, we aimed to provide the public with some feedback methods that could directly influence the exhibition's evolution. The main purpose of this is to catch the visitors' attention and to give them the feeling of being involved in the work's evolution.

Another difficulty has been to ensure that the program could also be easy to use for the artists, who are usually mostly unfamiliar with computer sciences. As the artists often have a precise idea of what their production should look like, they need to be able to understand a priori which global behavior would result from how they configure the system, at least to a certain extent. Hence, we had to link a media control system with a way for artists to express their preferences over the system behavior which should be easy to understand. The system should also be able to run in real-time, thus it should have a low computational complexity.

3.1 An agent-based solution

The main problem was to link media control and artists' wishes. We needed a system which could handle a complex network of media elements linked with one another and work autonomously with it. These requisites and the participative design of the exhibition between artists and scientists have led to the choice of a solution that has already been used in artistic works [4, 5]: a *multi-agent system*. Multi-agent systems offer the possibility of maintaining a structured organization through time, referring to laws and rules for their behavior while giving varying results, depending on environmental events and internal configurations. Unlike other AI paradigms, multi-agent systems allow the user to configure locally the agents' behavior, which is simple and efficient, without having to specify the overall system's behavior. Moreover, the agents' autonomy and their evolving organization allow the system to have a coherent behavior through time – as agents' behavior is specified – while never returning to the exact same configuration.

The first question when designing a multi-agent system concerns identifying the agents. The solution we adopted was taken for its simplicity and its understan-

dability for the artists more than for methodological reasons. It consists of associating each media – images, sounds or videos – with a single agent. This solution provides a way to distribute the control on the resulting work over local rules specific to sets of agents. Other solutions obviously exist, such as building a different agent for each simple behavior of the system, e.g. drawing, blinking, playing and looping. However, the chosen solution has the advantage of being easily understood by artists, who could manipulate sounds or images instead of unknown abstract entities.

3.2 VOWELS methodology

As the system design is based on agents and their interactions, and as we have already defined what the agents should be, the VOWELS methodology [2] is a useful approach for designing the system. Following this methodology, we have analyzed the problem in terms of Agents, Environments, Interaction and Organizations. For each of these terms, we can specify the considered entities, methods, the expected behaviors and the imposed implementation restrictions.

Since *agents* are associated with media, their attributes must include their type (sound or image), a name or reference to their associated media and their duration when applicable. For their behavior to be adaptable, they require some memory. We have chosen a simple form of memory as a table of associated keys and values (*e.g.* "active" = "yes"). In addition to being simple to manage it is expressively powerful. Since the agents' behaviors must be understandable and computationally simple, *reactive agents* are used. They allow complex emergent behaviors with low computational load and less complex inputs than cognitive agents – as some of the inputs should come from the artists, they need to be kept simple.

As agents were associated to media, *environments* were logically associated to displays (screens and speakers). Each environment represents a set of displays of various types, in which the agents' interactions occur. When interacting, an agent can play its media on the displays included in its current environment that correspond to the type of its media (*i.e.* screens for an image, speakers for a sound).

An *interaction* between two or more agents can only occur within an environment if all these agents are situated in this environment. We limit the interactions to take place inside environments because of the conceptual idea of proximity: situated agents can only interact when they perceive each other, meaning here that they are located in the same environment. Interactions may change the agents' memory, make them move, stop or resume some action. Interactions also trigger outputs, which take the form of user-defined strings, for the system to command the media application. We define an *interaction pattern* as a model of interactions that consists of triggering conditions and effects of the interaction. Each interaction is then an instantiation of an interaction pattern.

Finally, the system's *organization* is defined by two elements: the agents' repartition in the environments and the interaction patterns that are defined and ap-

plied. The important point is that organization is the only element that the users – either the artists or the public through interaction – can modify in the system. To do this, users must then be able to move agents from one environment to another, to activate or to deactivate them, and also to control which interaction patterns each agent will use. This control is applicable both for the initial situation, and then in real-time, in order to provide a sort of control script to the system or to modify the system's evolution through interaction with the public.

3.3 AGR interpretation

Another helpful approach for the system's design is the *Agent-Group-Role* paradigm [3], as it is a good complimentary approach to the VOWELS methodology. Following this method, we describe the system's organization along three dimensions: agents that compose the system; groups, which are sets of interacting agents; roles, which are functions' templates agents can adopt in the system.

Firstly, agents are defined as media elements representatives. Since agents can only interact inside environments, we can associate the notion of groups to sets of agents that are in the same environment. Then, defining roles consists in linking agents to functions in the system, which should be semantically understandable for the artists. For more simplicity, we assume that agents have fixed roles at their creation and cannot change their roles during their existence. If we need a media to be linked successively to two or more incompatible roles, we create two agents for the same media and link each one with a different role. Hence, role definition can be done by associating a set of *tags* – simple user-defined strings such as "slideshow image", "brief sound", "blue image", etc. – with each agent. These tags are then used in interaction patterns' formulation.

4 Software development: problems and solutions

Since the multi-agent system was to be part of a multimedia creation platform, Max/MSP/Jitter, there were many technical constraints. Firstly, the software had to be linked to an external API written in C, hence it had to be developed in C/C++ and then compiled as a dynamic library. Secondly, the software had to be usable on both Windows and Mac OS systems, because the software environment used by the artist was undefined at the time of development, and Max/MSP/Jitter exists on both operating systems. For that reason, the software had to be linked only with libraries that were available for both operating systems. Thirdly, only a few of all the possible agents would act at the same time, so we wanted to base their code on semaphores and monitors to minimize the computational load of the system; hence, we chose *asynchronous agents*.

Another problem has been the management of asynchronous inputs – commands given to the multi-agent system – and outputs – instructions passed from the system to Max/MSP/Jitter platform. Semaphores and monitors were sufficient to manage the distribution of inputs to appropriate agents or environments. Nevertheless, to give the system coherent orders – not passing a command to the platform and a contradictory one a split second later – we had to synchronize agents at two levels: environments and system. In each environment, agents must wait for every active one to propose an interaction before one is selected by highest priority. At the system level, commands to pass to the platform are added to a queue and sent out with a minimum delay in order not to saturate the platform capacity. This prevents the system from undertaking useless interactions (that occur but cannot produce a media output because the displays are already used by another interaction) and unnecessary computation. It is actually more efficient than a simple filter on the outputs because it requires less computation and because the delay it imposes on agents before they interact (waiting for others to compute their possible interactions) has no impact on the display, as it is only a few milliseconds long.

Another requirement is that the software configuration (entities and interactions' patterns) must be easy to define and to understand even for non-scientist digital artists, as it is the basis for all the exhibition's scripting. Hence, the system is initialized through two XML configuration files: one for agents, resources and environments and one for interactions' patterns. XML has been chosen because it is easily read and written by users. For scripting possibilities, users can move, activate and deactivate agents, and add or remove interactions' patterns to each agent in real-time, as defined earlier in the organizational description of the system.

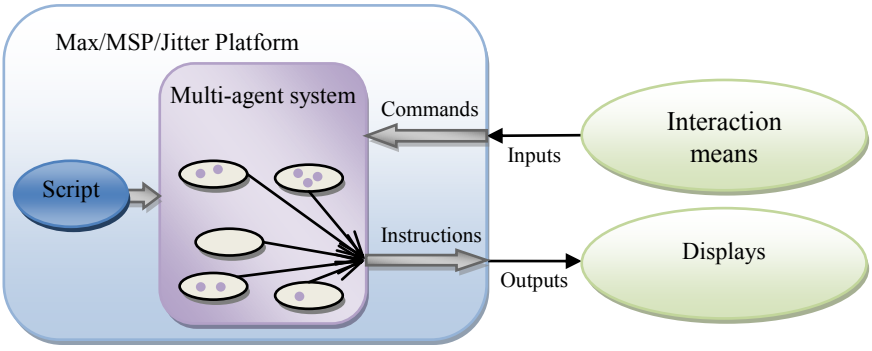

Fig. 1. System schema

Another major issue was to decide which behavior model should be chosen for the agents. The model has to be both simple to understand and efficient. It also needs to have the best expressive power possible, to allow more flexibility for the artists to choose the agents' behaviors. The agents have thus been given a *rule-based reasoning* capability [7]. This allows artists to easily understand the behaviors since the *interaction patterns* are coded by explicit rules. When agents interact,

they take the needed resources for a given time, send commands to Max/MSP/Jitter platform and apply effects that are internal to the multi-agent system; then they release resources and resume checking for other applicable rules. A rule could be described by a sentence such as "WHEN (memory state "active" = "yes") and (a partner with tag "image" is present) THEN (set memory state "active" to "no") and (play sound for 5ms) and (display partner's image for 5ms)".

Rules are defined as a composition of: a set of conditions, a set of required resources, a set of effects to apply to agents in the system and a set of commands to pass to the media platform. Conditions are defined relative to the presence of a type (particular tag, meaning particular role) of agent or the value associated to a particular key in the agent's memory, which can be combined through Boolean operators. Effects (moving, activating, deactivating or changing memory values), and commands (user-defined character strings with wild-cards) can be delayed for a value between zero, meaning immediate, and the interaction's duration.

In order to compensate for the situations when rule-based reasoning could not provide the agents with the way to behave as expected (*e.g.* if an interaction through two random agents should occur only once in each environment, or only in one environment at a time), we have added the possibility of defining agents that are not linked to media. These agents are just considered as triggers for particular events, and can act as such when given the appropriate behavior rules.

5 Resulting software and exhibition

For the exhibition, the artists used three kinds of media: photographs of the campus, photographs of visitors to the exhibition, and sounds they had created. The hardware installation was composed of 3 computers, running the multi-agent system and controlling the displays, a buzzer and a keyboard for visitors' interaction, a camera to take photographs of the visitors when they entered the room, a mini-printer, 4 speakers and 6 LCD screens, one of them displaying a view of interactions taking place in the system and the others displaying the media.

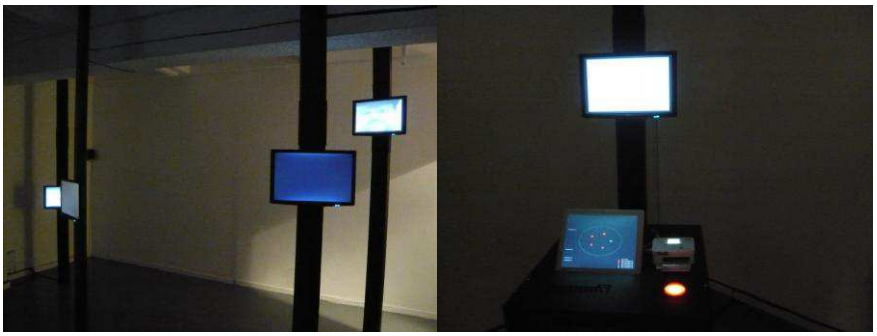

Fig. 2.a and 2.b The exhibition setup

Due to the number of screens available for media display, the software settings involved 5 environments, each corresponding to a screen and sharing all the speakers. The configuration also involved more than 500 agents, mostly linked to photographs, but with a few sounds and triggers. Only a maximum of 250 agents were activated simultaneously in the artists' script because of artistic needs.

About 30 different rules were defined in this configuration and used during the 4 steps of the script written by the artists. Approximately 30 different tags were used to describe the roles. During the first step, two agents (among about 40) representing groups of images interacted with a trigger to be selected. Then, image agents from the groups they represent interacted in pairs on each screen to be displayed simultaneously by applying some textural transformations on the images. During the second step, triggered by a timer, sounds interacted with images to display quickly the images on a screen while playing the sound, in order to give the feeling of an accelerating explosion of sounds and images. During the third step, also started by a timer, agents representing groups of images interact to trigger slideshows displays on each screen. Then, during the last step, triggered when anyone from the public presses the buzzer, an image agent, from a selection of about 200, interacts with a trigger to display an image. A member of the public has to enter a title for the image before it fades to its mean color, and a slideshow involving all titled images is displayed together with a representation of the set of all titles. Then the displayed single-colored picture and its associated keyword are printed as a summary of the visit (to the artists' point of view) and some rules are applied to the agents to reset some step-specific memory for another cycle.

Fig. 3.a and 3.b Representation of the MAS's interactions and picture of the exhibition

6 Evaluation

As a result of the exhibition, we were able to evaluate if the software met its objectives or not. First of all, the system was able to run correctly in real-time on a dual core computer with a configuration of more than 200 agents acting at the

same time. Whereas the system has not been tested with more than 500 agents, it seems that, on a reasonably powerful computer – a 3GHz quad-core – the program runs smoothly; so we can assert that the system is able to be used with enough agents for most artists' needs on a common computer.

Other evaluations resulted from questioning and interviewing the artists and the public (amongst were which other digital artists, able to evaluate more accurately the artistic work). Firstly, as far as we could see, the agent paradigm was easy to understand because of the association between agents and media, and the tagging system to define roles was also very simple for artists. However, devising the rules, and especially defining priorities and conditions was something theoretically hard to understand, and the loss of control over precise behavior of the system induced by its autonomy was something difficult to accept in the first place by artists. Manipulation of XML was not a major difficulty though, given a few sample files and explanation, even for non-specialist artists.

Additionally, we observed that the public found it difficult to understand the meaning of the interactions and the role of the multi-agent system despite the artists emphasizing the agent paradigm by displaying interactions on a separate screen and by giving the public a small written explanation of the project prior to the visit. Nevertheless, the exhibition generated good feedback from the public as well as the theatre representatives. Amongst the good points that were mentioned was interactivity. People saw themselves as part of the exhibition because it incorporated photographs of them and because they were invited to buzz at some time and enter words on the keyboard. Another mentioned good point was the non-repetitive displays due to the agents never having exactly the same configuration.

7 Conclusions and perspectives

For this project, we have constructed a software component based on the multi-agent paradigm and integrated it into a media creation and control platform. This system allows artists, given a few explanations on how to construct rules, to easily build and script a dynamic exhibition based on sound and video display. Once constructed, this exhibition can be left on its own to evolve autonomously through time and react interactively to public presence or actions.

From the artists' and public's feedbacks, we can determine that the software component has been useful for artists. We can see that it allows them to create interesting possibilities of interaction and autonomy, which is something that the public would notice and appreciate. However, we can also see that, despite given explanations, the scientific part of the project is hard to understand for the public.

A first possible improvement concerns the positioning of agents. Currently there is no definition of a position for agents in the environments. That means that the only location we define for an agent is the actual environment in which it is situated. Adding some coordinates and maybe an environment topology could be a good way to improve the display abilities.

Another way to improve the system, regarding its interactivity, would be to change the way the public is considered. Currently, the public has the same abilities as the artists, i.e. they can add and remove agents and rules. However, they are limited by the interactions means, especially in this exhibition, where they could only interact by triggering portions of scripts that had been designed by the artists (i.e. they did not actually choose the modifications they wanted to apply to the system but only triggered those that have been coded by the artists). A powerful improvement would be to consider the public as one or more agents so they can interact with other agents by triggering rules, as agents interact with each other.

Finally, this work may be extended by integrating it into a larger piece of software concerned with assisting creativity [6]. The use of XML in the system makes this possibility easier, because it allows the software to smoothly interact with any potential creativity assistant (which could just manipulate the XML configuration files). This could allow artists to use intelligent assistants to help them defining their agents, the tags and the rules to obtain the configuration that fit their views the most.

Acknowledgments We would like to acknowledge for their participation in the project the Grenoble theatre Hexagone, in particular Cécile Gauthier; the "Service Culturel de Grenoble Universités", specifically Bertrand Vignon; the members of Coincoin Production: Julien Castet, Mélanie Cornillac, Emilie Darroux and Maxime Houot; and of course Antoine Lefebvre, the engineering computer science student at ENSIMAG who programmed the multi-agent system.

References

1. H. Cohen, "How to draw three people in a botanical garden," *AAAI*, vol. 89, pp. 846-855, 1988.
2. Y. Demazeau, "From Cognitive Interactions to Collective Behaviour in Agent-Based Systems", 1st European Conference on Cognitive Science, Saint-Malo, France, pp. 117-132, 1995.
3. J. Ferber, O. Gutknecht, F. Michel, "From Agents to Organizations: an Organizational View of Multi-Agent Systems", Agent-Oriented Software Engineering (AOSE) IV, P. Giorgini, Jörg Müller, James Odell eds., Melbourne, July 2003, LNCS 2935, pp. 214-230, 2004.
4. Y. Gufflet and Y. Demazeau, "Applying the PACO paradigm to a three-dimensional artistic creation", 5th International Workshop on Agent-Based Simulation, ABS'04, pp. 121-126, SCS, Lisbon, Portugal, 2004.
5. G. Hutzler, B. Gortais, A. Drogoul, "Le Jardin Des Hasards: peinture abstraite et Intelligence Artificielle Distribuée réactive", in Journées Francophones d'Intelligence Artificielle Distribuée et Systèmes Multi-Agents, La Colle sur Loup, J.-P. Muller and J. Quinqueton, eds., pp. 295-306, Hermès, Paris, 1997.
6. U. Lösch, J. Dugdale, Y. Demazeau, "Requirements for Supporting Individual Human Creativity in the Design Domain", in *Proceedings of the 8th international Conference on Entertainment Computing* (Paris, France, September 03 - 05, 2009), S. Natkin and J. Dupire, eds., Lecture Notes In Computer Science, vol. 5709, Springer-Verlag, Berlin, Heidelberg, pp. 210-215, 2009.
7. J. McDermott, "R1: A Rule-Based Configurer of Computer Systems", *Artificial Intelligence*, Vol.19, No 1, pp. 39-88, 1982